Program Verification
An Application of Predicate Logic
Program verification

- Introduction and Hoare triples
- Inference rules for verification
- Loop invariants
The Idea

• Verification:
  – (Does the program run and terminate?)
  – Does it accomplish its goal? (partially correct)

• Use logic to:
  – Specify the initial state of a program
  – Express how statements change states
  – Specify the final state = goal of the program
  – Prove that we can…start at the start, go to the goal, stop
Generic Statements

Simple statement types to show the idea:

- Assignment
- Sequencing
- If-then-else
- While- do
Hoare Triple

• A proposition about a program statement
• Specifies the changes in what is true
• General Form
  – \( p \) \{\( S \)\} \( q \)
  – \( p \) and \( q \) are propositions
  – \( S \) is a program statement
• If \( p \) is true
• and statement \( S \) is executed
• then \( q \) is true after execution, assuming \( S \) terminates
• \( p \) is the precondition and \( q \) is the post condition.
• \( S \) could be a single statement or an entire program
Inference Rule for Verification

Assignment

\[ p(e) \{ v \leftarrow e \} p(v) \]

- \( \leftarrow \) is the assignment operator
- If \( e \) satisfies predicate \( p \) before execution, then \( v \) satisfies \( p \) afterward.
- Examples:
  - \( \text{Odd}(3) \{ x \leftarrow 3 \} \text{Odd}(x) \)
  - \( \text{Odd}(y) \{ x \leftarrow y+2 \} \text{Odd}(x) \)
  - \( \text{Odd}(x) \{ x \leftarrow x+1 \} \text{Even}(x) \)
    - \( X_{\text{after}} = X_{\text{before}} + 1 \)
Inference Rules for Verification

Sequence

\[ p \{S1\} q \]
\[ q \{S2\} r \]
\[ p \{S1, S2\} r \]

- \( q \): postcondition for \( S1 \) and precondition for \( S2 \)
- Example

\[
(x = 1) \quad \{y \sqsupset 3\} \quad (x=1) \sqsupset (y=3)
(x=1) \sqsupset (y=3) \quad \{z \sqsupset x + y\} \quad (z=4)
(x=1) \quad \{y \sqsupset 3; z \sqsupset x + y\} \quad (z = 4)
\]
Another Example

(x = y) \{x \sqcap x+1\} (x = y+1)
(x = y+1) \{y \sqcap y+1\} (x = y)
(x = y) \{x \sqcap x+1; y \sqcap y+1\} (x = y)

- x = y \text{ “Invariant”}
  - x=y holds on entry and it holds on exit
  - x \neq y is possible during execution
  - Subscripts avoid confusion
Subscripts

\[ x_0 = y_0 \]
\[ x_1 = x_0 + 1 \quad \text{and} \quad y_1 = y_0 \quad \text{step 1 - statement 1} \]
\[ y_2 = y_1 + 1 \quad \text{and} \quad x_2 = x_1 \quad \text{step 2 - statement 2} \]
\[ x_2 = x_1 = x_0 + 1 = y_0 + 1 = y_1 + 1 = y_2 \]
\[ x_2 = y_2 \]
Inference Rules for Verification

Conditionals

If-then

\[(p \land B) \{S\} (q)\]
\[(p \land \neg B) \{q\}\]
\[p \{\text{if } B \text{ then } S\} q\]

- If \( p \) is true then either
  - If \( B \) is True \( q \) becomes true when \( S \) executes or
  - If \( B \) is False \( q \) is already True (provable)

If-then-else

\[(p \land B) \{S1\} (q)\]
\[(p \land \neg B) \{S2\} (q)\]
\[p \{\text{if } B \text{ then } S1 \text{ else } S2\} q\]
Example

\[(p \triangleright B) \{S\} (q)\]
\[(p \triangleright \neg B) \triangleright (q)\]
p {if B then S} q

- Statement: \{if \(y < x\) then \(y \triangleleft x\)\}
- Precondition \(x = 7\)
- Show the postcondition \(y \geq 7\)
- \(B \equiv y < x\) so \(\Box B \equiv y \geq x\)
- \(S \equiv y \triangleleft x\)

\[
(x = 7 \triangleright y < x) \quad \{y \triangleleft x\} \quad (y \geq 7) \\
(x = 7 \triangleright y \geq x) \quad \triangleright \quad (y \geq 7) \\
x = 7 \quad \{\text{if } y < x \text{ then } y \triangleleft x\} \quad (y \geq 7)
\]
Verify Code for |X|

\[(p \land B) \{S_1\} (q)\]
\[(p \land \neg B) \{S_2\} (q)\]
\[p \{\text{if } B \text{ then } S_1 \text{ else } S_2\} q\]

\[(\text{TRUE } \land x < 0) \{\text{abs } \land -x\} \quad \text{(abs = |x|)}\]
\[(\text{TRUE } \land x \geq 0) \{\text{abs } \land x\} \quad \text{(abs = |x|)}\]
\[(\text{TRUE}) \{\text{if } x < 0 \text{ then abs } \land -x \text{ else abs } \land x\} \quad \text{(abs = |x|)}\]
Loop Invariants

Example

While:

\[(p \land B) \{S\} p\]
\[p \{\text{while } B \text{ do } S\} (p \land \lnot B)\]

• **The oddball puzzle:**
• A bag has M white and N black balls.
• You repeatedly randomly remove two.
• If opposite color, put the white one back.
• If same color, put a black one back (you must have extras).
• Eventually one ball is left since we remove one ball on each turn
• What color is it?
Selections

- 17W 10B: draw 1W & 1B then return W
- 17W 9B: draw 2B then return B
- 17W 8B: draw 2W the return B
- 15W 9B
- ....
Solution Using Loop Invariant

- $S =$ the entire code of the selection, replacement counter update loop.

while $(m+n > 1)$ do
    choose a ball - if white decrement white count
    else decrement black count.

    choose a ball - if white decrement white count
    else decrement black count

    if the choices match increment black count
    else increment the white count
Applying the Inference Rulee

- \( p = \) the loop invariant is \( (\text{odd(white count)}) \)
  - which must equal \( \text{odd(WHITECOUNT)} \).
  - Exhaust the possible cases
    - If both are black, we return a black and the parity of white is unchanged.
    - If both are white, we return a black and the parity of white remains the same as it was before choosing (although the count is reduced by two).
    - If we choose one of each, we return the white and the parity remains unchanged.
  - \( B = \) condition for staying in loop, \( m+n > 1 \).
  - \( p \{\text{while B do S}\} (p \square \square B) \)
  - One ball is left and the parity of the white count is the same as it was when we started
    - Since we started with an odd number of white balls, we must have a white ball left to guarantee the parity is preserved.
Another Example for $n!$

\[(p \sqcap B) \{S\} p \sqsubseteq\]
\[p \{\text{while } B \text{ do } S\} \ (p \sqcap \neg B)\]

- Computing $n!$
  
  \[
i \sqsubseteq 1;
  \]
  \[
f \sqsubseteq 1;
  \]
  \[
  \text{while } i < n \text{ do }
  \]
  
  \[
  \text{begin }
  \]
  \[
i \sqsubseteq i+1;
  \]
  \[
f \sqsubseteq f \ast i;
  \]
  
  \[
  \text{end }
  \]
• Compute $n!$ in the variable $f$.
• The loop invariant includes $f = i!$ at the beginning of the loop,
• Also, make $i=n$ at loop exit, so that $f = n!$.
• Entire loop invariant $p : (f = i!) \land (i \leq n)$.
• Must have $n \geq 1$ initially, to make $p$ true.
• $p$ is a loop invariant, with loop condition, $B = (i < n)$ if:
  
  \begin{align*}
  & (f = i!) \land (i \leq n) \land (i < n) \\
  \{ & i \rightarrow i+1; f \rightarrow f \times i \} \\
  & (f = i!) \land (i \leq n) \land (i < n)
  \end{align*}
What the Rule Gives Now

\[(f = i!) \land (i \leq n)\]

\{while(i < n) do S\}

\[(f = i!) \land (i \leq n) \land (i \geq n)\]

... so when the loop terminates: \(i = n\), and so \(f = n!\).
Summary

• Introduction and Hoare triples
• Inference rules for verification
  – Assignment
  – Sequencing
  – If-then-else
  – While- do
• Loop invariants